The Midterm
1. List one kind of mutable data type, and one type of immutable data type, and describe the difference between immutable and mutable

- **Mutable**: list, dict, int, float  Can change/update
- **Immutable**: string, tuple  Can’t be changed
2. What is the name of the method that allows you access to a dictionary’s key, value pairs?

- `.items()`
4. What class method allows you to define a pretty output for print on a user-defined class?

- Either `__str__` or `__repr__`
6. Write the code to determine if a piece of Python code was run from the command line (versus, say, a module?)

- if ‘name’ == __main__:
9. Write all execution output for the following programs, and if the code is not correct and would produce and error, describe the error:

(a) \[A = ("red", "green", "blue")\]
print(A[1]) \hspace{1cm} \text{green}
A[2] = "orange" \hspace{1cm} \text{can’t update tuple error}
print(A[2])

(b) \[L = [1, 2, 3, 4]\]
print(“The last element of this list is”, L[len(L)])
\hspace{1cm} \text{out of range error}
(c) 1stVariable = 3.14
print 1stVariable + int(10)

(d) two = 3
my_dict = {"one": two, "two": "3", "three": 1}
print (my_dict["three"] – my_dict["one"] * int(my_dict["two"]))  # -8
print (my_dict[0])  # Key error (no 0 key in dict)

(e) myList = ["cat", "dog", "TAKE A DEEP BREATH! ", "pig", "cow"]
print (myList[-1])  # cow
print (myList[:3])  # ['cat', 'dog', 'TAKE A DEEP BREATH! ']
param = "blah"
def my_func(param):
    param = param + "blah"
    print ("Your parameter was: param")
    return param
print (my_func(param))  # Your parameter was: param blahblah
print (my_func("param2"))  # Your parameter was: param param2blah
print (my_func(blah))  # Error: Name blah is not defined
The following functions are silly error checking to make sure two variables are in the proper range for day and month (NOTE: No type checking is done, and that’s okay). A sample invocation is also included. Fill in the blanks so the program would execute like you would expect. There are EIGHT blanks.

```python
def check_month(month):
    if (______________ > 0) and (______________ < 13):
        return True
    _____________:
    return False  month, month, else
```
def check_day(day):
    if (day < 0) or (day > 31):
        return __________________
    else:
        return __________________
    or, False, True

def check_inputs(day, month):
    try:
        valid = check_month(month) and check_day(day)
    __________________:
        valid = False

    if valid:
        print ("Your inputs are valid")
    else:
        print ("Your inputs are invalid")
and, except
11. Consider the following variable `d`, which is a dictionary, with keys of integers and values of lists (of integers):

For example: `d = {0:[1,2,3], 3:[2,3,4,0], -1:[-1,3,4]}`

Write a Python function `sum_lists` that takes a dictionary argument such as `d` and returns a new dictionary with the same keys, but the values are the sums of the elements in the lists. For example, if `d` is the argument, `sum_lists` should return:

```
summed_list = {0:6, 3:9, -1:6}
```

The header for the function is: `def sum_lists(dict_of_lists)`.

```
def sum_lists(dict_of_lists):
    result = {}
    for k in dict_of_lists:
        result[k]=sum(dict_of_lists[k])
    return result
```
12. List one potential data type(s) that the following function accepts as input

def f(a):
    count = 0
    (oldx, oldy, oldz) = (1, 1, 1)
    for (x, y, z) in a:
        if ((x != oldy) or (y != oldz)):
            return 42
        elif (x + y != z):
            return -42
        else:
            (oldx, oldy, oldz) = (x, y, z)
            count += 1
    return count

• {List or tuple} of {a combination of 3-element lists or tuples}
13. What would you give as input to the function `f` to return the value of `count` as the integer 3?

```python
def f(a):
    count = 0
    (oldx, oldy, oldz) = (1,1,1)
    for (x,y,z) in a:
        if ((x != oldy) or (y != oldz)):
            return 42
        elif (x+y != z):
            return -42
        else:
            (oldx, oldy, oldz) = (x,y,z)
            count += 1
    return count
```

f(((1,1,2),(1,2,3),(2,3,5)))
f(((1,1,2),(1,2,3),(2,3,5)))
f(((1,1,2),(1,2,3),(2,3,5)))
f(((1,1,2),(1,2,3),(2,3,5)))
f(((1,1,2),(1,2,3),(2,3,5)))
and more…
Networked Programs
Transport Control Protocol (TCP)

- Built on top of IP (Internet Protocol)
- Assumes IP might lose some data - stores and retransmits data if it seems to be lost
- Handles “flow control” using a transmit window
- Provides a nice reliable pipe

TCP Connections / Sockets

“In computer networking, an Internet socket or network socket is an endpoint of a bidirectional inter-process communication flow across an Internet Protocol-based computer network, such as the Internet.”

http://en.wikipedia.org/wiki/Internet_socket
TCP Port Numbers

- A port is an application-specific or process-specific software communications endpoint
- It allows multiple networked applications to coexist on the same server
- There is a list of well-known TCP port numbers

http://en.wikipedia.org/wiki/TCP_and_UDP_port
www.umich.edu

Incoming E-Mail

Login

Web Server

Personal Mail Box

25

23

80

443

109

110

74.208.28.177

blah blah blah

Clipart: http://www.clker.com/search/networksym/1
Common TCP Ports

- Telnet (23) - Login
- SSH (22) - Secure Login
- HTTP (80)
- HTTPS (443) - Secure
- SMTP (25) (Mail)
- IMAP (143/220/993) - Mail Retrieval
- POP (109/110) - Mail Retrieval
- DNS (53) - Domain Name
- FTP (21) - File Transfer

Sometimes we see the port number in the URL if the web server is running on a “non-standard” port.
Sockets in Python

Python has built-in support for TCP Sockets

```python
import socket
mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect( ('data.pr4e.org', 80) )
```

http://docs.python.org/library/socket.html
Application Protocols
Application Protocol

• Since TCP (and Python) gives us a reliable socket, what do we want to do with the socket? What problem do we want to solve?

• Application Protocols
  
  - Mail
  
  - World Wide Web

HTTP - Hypertext Transfer Protocol

- The dominant Application Layer Protocol on the Internet
- Invented for the Web - to Retrieve HTML, Images, Documents, etc.
- Extended to be data in addition to documents - RSS, Web Services, etc. Basic Concept - Make a Connection - Request a document - Retrieve the Document - Close the Connection

http://en.wikipedia.org/wiki/Http
HTTP

The HyperText Transfer Protocol is the set of rules to allow browsers to retrieve web documents from servers over the Internet.
What is a Protocol?

• A set of rules that all parties follow so we can predict each other’s behavior

• And not bump into each other

  - On two-way roads in USA, drive on the right-hand side of the road

  - On two-way roads in the UK, drive on the left-hand side of the road
http://www.dr-chuck.com/page1.htm

protocol  host  document

http://www.youtube.com/watch?v=x2GylLq59rl
1:17 - 2:19

https://www.youtube.com/watch?v=x2GylLq59rl&t=1m17s
Getting Data From The Server

• Each time the user clicks on an anchor tag with an href= value to switch to a new page, the browser makes a connection to the web server and issues a “GET” request - to GET the content of the page at the specified URL

• The server returns the HTML document to the browser, which formats and displays the document to the user
Web Server

80

Browser

The First Page

If you like, you can switch to the Second Page.
GET http://www.dr-chuck.com/page2.htm

The First Page
If you like, you can switch to the Second Page.
GET http://www.dr-chuck.com/page2.htm

<h1>The Second Page</h1><p>If you like, you can switch back to the <a href="page1.htm">First Page</a>.</p>
If you like, you can switch back to the [First Page](http://www.dr-chuck.com/page1.htm).

---

**Request**

GET http://www.dr-chuck.com/page2.htm

**Web Server**

![Diagram of web server](image)

80

**Response**

<h1>The Second Page</h1>

If you like, you can switch back to the [First Page](http://www.dr-chuck.com/page1.htm).

**Browser**

![Diagram of browser](image)

The First Page

If you like, you can switch to the [Second Page](http://www.dr-chuck.com/page2.htm).

---

The Second Page

If you like, you can switch back to the [First Page](http://www.dr-chuck.com/page1.htm).
Internet Standards

• The standards for all of the Internet protocols (inner workings) are developed by an organization

• Internet Engineering Task Force (IETF)

• www.ietf.org

• Standards are called “RFCs” - “Request for Comments”

Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information...
5 Request

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

\[
\text{Request} \quad = \quad \text{Request-Line} \quad ; \quad \text{Section 5.1}
\]

\[\begin{align*}
* & (( \quad \text{general-header} \quad ; \quad \text{Section 4.5} \\
| & \quad \text{request-header} \quad ; \quad \text{Section 5.3} \\
| & \quad \text{entity-header} \quad ) \quad \text{CRLF}) \quad ; \quad \text{Section 7.1} \\
\text{CRLF} & \quad [ \quad \text{message-body} \quad ] \quad ; \quad \text{Section 4.3}
\end{align*}\]

5.1 Request-Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

\[
\text{Request-Line} \quad = \quad \text{Method SP Request-URI SP HTTP-Version CRLF}
\]
Making an HTTP request

• Connect to the server like www.dr-chuck.com

• Request a document (or the default document)

  • GET http://www.dr-chuck.com/page1.htm HTTP/1.0
  • GET http://www.mlive.com/ann-arbor/ HTTP/1.0
  • GET http://www.facebook.com HTTP/1.0
  • GET http://jeremyabramson.com HTTP/1.0
$ telnet http://jeremyabramson.com/index.html 80
Trying 208.85.3.226...
Connected to jeremyabramson.com.
Escape character is '^]'.
GET http://jeremyabramson.com/index.html HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 05 Mar 2018 06:44:26 GMT
Server: Apache/2
Upgrade: h2,h2c
Connection: Upgrade, close
Last-Modified: Tue, 19 Jul 2016 18:27:15 GMT
ETag: "401-538013bec06c0"
Accept-Ranges: bytes
Content-Length: 1025
Vary: Accept-Encoding,User-Agent
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>jeremyabramson.com -- The Internet home of Jeremy Abramson. Coming soon*!</title>
</head>
<body>
  <h2>Welcome to the Internet home of Jeremy Abramson, Ph.D.
  . . . .
  Connection closed by foreign host. 45
</h2>
Let’s Write a Web Browser!
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\r\n\r\n'.encode()
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if (len(data) < 1):
        break
    print(data.decode(), end='')
mysock.close()
HTTP/1.1 200 OK
Date: Sun, 14 Mar 2010 23:52:41 GMT
Server: Apache
Last-Modified: Tue, 29 Dec 2009 01:31:22 GMT
ETag: "143c1b33-a7-4b395bea"
Accept-Ranges: bytes
Content-Length: 167
Connection: close
Content-Type: text/plain

But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

while True:
    data = mysock.recv(512)
    if ( len(data) < 1 ) :
        break
    print(data.decode())
About Characters and Strings...
ASCII
American Standard Code for Information Interchange

https://en.wikipedia.org/wiki/ASCII
http://www.catonmat.net/download/ascii-cheat-sheet.png
Representing Simple Strings

• Each character is represented by a number between 0 and 256 stored in 8 bits of memory

• We refer to "8 bits of memory as a byte" of memory – (i.e. my disk drive contains 3 Terabytes of memory)

• The ord() function tells us the numeric value of a simple ASCII character

  >>> print(ord('H'))
  72
  >>> print(ord('e'))
  101
  >>> print(ord('\n'))
  10

Check it out!
In the 1960s and 1970s, we just assumed that one byte was one character
## Scripts

### European Scripts
- Armenian
- Armenian Ligatures
- Caucasian Albanian
- Cypriot Syllabary
- Cyrillic
  - Cyrillic Supplement
  - Cyrillic Extended-A
  - Cyrillic Extended-B
  - Cyrillic Extended-C
- Elbasan
- Georgian
  - Georgian Supplement
- Glagolitic
  - Glagolitic Supplement
- Gothic
- Greek
  - Greek Extended
  - Ancient Greek Numbers
- Latin
  - Basic Latin (ASCII)
  - Latin-1 Supplement
  - Latin Extended-A
- Middle Eastern Scripts
  - Anatolian Hieroglyphs

### African Scripts
- Adlam
- Bamum
- Bassa Vah
- Coptic
  - Coptic in Greek block
  - Coptic Epact Numbers
- Egyptian Hieroglyphs (1MB)
- Ethiopic
  - Ethiopic Supplement
  - Ethiopic Extended
  - Ethiopic Extended-A
- Mende Kikakui
  - Meroitic
  - Meroitic Cursive
  - Meroitic Hieroglyphs

### South Asian Scripts
- Ahom
- Bengali and Assamese
- Bhaiksuki
- Brahmi
- Chakma
- Devanagari
  - Devanagari Extended
- Grantha
- Gujarati
- Gurumuki
- Kaithi
- Kannada
- Kharoshthi
- Khojki
- Khudawadi
- Lepcha
- Limbu
- Mahajani
- Malayalam
- Meetei Mayek
  - Meetei Mayek Extensions
- Modi

### Indonesia & Oceania Scripts
- Balinese
- Batak
- Buginese
- Buhid
- Hanunoo
- Javanese
- Rejang
- Sundanese
  - Sundanese Supplement
- Tagalog
- Tagbanwa

### East Asian Scripts
- Bopomofo
  - Bopomofo Extended
- CJK Unified Ideographs (Han) (35MB)
  - CJK Extension-A (6MB)
  - CJK Extension B (40MB)
  - CJK Extension C (3MB)
  - CJK Extension D
  - CJK Extension E (3.5MB)
    - (see also Unihan Database)
- CJK Compatibility Ideographs
Multi-Byte Characters

To represent the wide range of characters computers must handle we represent characters with more than one byte

- **UTF-16** – Fixed length - Two bytes
- **UTF-32** – Fixed Length - Four Bytes
- **UTF-8** – 1-4 bytes
  - Upwards compatible with ASCII
  - Automatic detection between ASCII and UTF-8
  - UTF-8 is recommended practice for encoding data to be exchanged between systems

*Unicode = Universal Coded Character Set*

*UTF = Unicode Transformation Format – 8-bit*
Two Kinds of Strings in Python

Python 2.7.10
>>> x = '이광춘'
>>> type(x)
<type 'str'>
>>> x = u'이광춘'
>>> type(x)
<type 'unicode'>

Python 3.5.1
>>> x = '이광춘'
>>> type(x)
<class 'str'>
>>> x = u'이광춘'
>>> type(x)
<class 'str'>

In Python 3, all strings are Unicode
Python 2 versus Python 3

Python 2.7.10
>>> x = b'abc'
>>> type(x)
<type 'str'>
>>> x = '이광춘'
>>> type(x)
<type 'str'>
>>> x = u'이광춘'
>>> type(x)
<type 'unicode'>

Python 3.5.1
>>> x = b'abc'
>>> type(x)
<class 'bytes'>
>>> x = '이광춘'
>>> type(x)
<class 'str'>
>>> x = u'이광춘'
>>> type(x)
<class 'str'>
Python 3 and Unicode

• In Python 3, all strings internally are UNICODE

• Working with string variables in Python programs and reading data from files usually "just works"

• When we talk to a network resource using sockets or talk to a database we have to encode and decode data (usually to UTF-8)

Python 3.5.1
>>> x = b'abc'
>>> type(x)
<class 'bytes'>
>>> x = '이광춘'
>>> type(x)
<class 'str'>
>>> u'이광춘'
>>> type(x)
<class 'str'>
Python Strings to Bytes

- When we talk to an external resource like a network socket we send bytes, so we need to encode Python 3 strings into a given character encoding.

- When we read data from an external resource, we must decode it based on the character set so it is properly represented in Python 3 as a string.

```python
while True:
    data = mysock.recv(512)
    if ( len(data) < 1 ) :
        break
    mystring = data.decode()
    print(mystring)
```
An HTTP Request in Python

```python
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\n\n'.encode()
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if (len(data) < 1):
        break
    print(data.decode())
mysock.close()
```
bytes.decode(encoding="utf-8", errors="strict")
bytearray.decode(encoding="utf-8", errors="strict")

Return a string decoded from the given bytes. Default encoding is 'utf-8'. errors may be given to set a different error handling scheme. The default for errors is 'strict', meaning that encoding errors raise a UnicodeError. Other possible values are 'ignore', 'replace' and any other name registered via codecs.register_error(), see section Error Handlers. For a list of possible encodings, see section Standard Encodings.

str.encode(encoding="utf-8", errors="strict")

Return an encoded version of the string as a bytes object. Default encoding is 'utf-8'. errors may be given to set a different error handling scheme. The default for errors is 'strict', meaning that encoding errors raise a UnicodeError. Other possible values are 'ignore', 'replace', 'xmlcharrefreplace', 'backslashreplace' and any other name registered via codecs.register_error(), see section Error Handlers. For a list of possible encodings, see section Standard Encodings.

https://docs.python.org/3/library/stdtypes.html#bytes.decode
https://docs.python.org/3/library/stdtypes.html#str.encode
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))

# Convert command to bytes
cmd = 'GET http://data.pr4e.org/romeo.txt HTTP/1.0\n\n'.encode()
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if (len(data) < 1):
        break
    print(data.decode())
mysock.close()
Making HTTP Easier With urllib
Using **urllib** in Python

Since HTTP is so common, we have a library that does all the socket work for us and makes web pages look like a file

```python
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
for line in fhand:
    print(line.decode().strip())
```

*urllib1.py*

Run the code!
But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

import urllib.request, urllib.parse, urllib.error
fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
for line in fhand:
    print(line.decode().strip())
Like a File...

```python
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')

counts = dict()
for line in fhand:
    words = line.decode().split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

print(counts)
```

urlwords.py
Reading Web Pages

```python
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://www.dr-chuck.com/page1.htm')
for line in fhand:
    print(line.decode().strip())

<h1>The First Page</h1>
<p>If you like, you can switch to the <a href="http://www.dr-chuck.com/page2.htm">Second Page</a>.</p>
```

urllib2.py
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://www.dr-chuck.com/page1.htm')
for line in fhand:
    print(line.decode().strip())

<h1>The First Page</h1>
<p>If you like, you can switch to the <a href="http://www.dr-chuck.com/page2.htm">Second Page</a>.</p>
import urllib.request, urllib.parse, urllib.error

fhand = urllib.request.urlopen('http://www.dr-chuck.com/page1.htm')
for line in fhand:
    print(line.decode().strip())
Parsing HTML
(a.k.a. Web Scraping)
What is Web Scraping?

- When a program or script pretends to be a browser and retrieves web pages, looks at those web pages, extracts information, and then looks at more web pages

- Search engines scrape web pages - we call this “spidering the web” or “web crawling”

http://en.wikipedia.org/wiki/Web_scraping
http://en.wikipedia.org/wiki/Web_crawler
Why Scrape?

- Pull data - particularly social data - who links to who?
- Get your own data back out of some system that has no “export capability”
- Monitor a site for new information
- Spider the web to make a database for a search engine
Scraping Web Pages

- There is some controversy about web page scraping and some sites are a bit snippy about it.
- Republishing copyrighted information is not allowed
- Violating terms of service is not allowed
The Easy Way - Beautiful Soup

- You could do string searches the hard way
- Or use the free software library called BeautifulSoup from www.crummy.com

https://www.crummy.com/software/BeautifulSoup/
BeautifulSoup Installation

# To run this, you can install BeautifulSoup
# https://pypi.python.org/pypi/beautifulsoups4

# Or download the file
# http://www.py4e.com/code3/bs4.zip
# and unzip it in the same directory as this file

import urllib.request, urllib.parse, urllib.error
from bs4 import BeautifulSoup

...
import urllib.request, urllib.parse, urllib.error
from bs4 import BeautifulSoup

url = input('Enter - ')
html = urllib.request.urlopen(url).read()
soup = BeautifulSoup(html, 'html.parser')

# Retrieve all of the anchor tags
tags = soup('a')
for tag in tags:
    print(tag.get('href', None))

python urllinks.py
Enter - http://www.dr-chuck.com/page1.htm
http://www.dr-chuck.com/page2.htm

Run the code!
Good BeautifulSoup Tutorial

- http://web.stanford.edu/~zlotnick/TextAsData/Web_Scraping_with_Beautiful_Soup.html
- Also, the Beautiful Soup Documentation is very good
Summary

- The TCP/IP gives us pipes / sockets between applications
- We designed application protocols to make use of these pipes
- HyperText Transfer Protocol (HTTP) is a simple yet powerful protocol
- Python has good support for sockets, HTTP, and HTML parsing